REMARKS

Reconsideration of the application is respectfully requested in view of the foregoing amendments and following remarks. Claims 27-49, 71-79, 81-83, and new claims 84-89 are pending in the application. No claims have been allowed. Claims 71, 78, and 81 are independent.

Objection to the Drawings by the Draftsperson

Applicants acknowledge the objection to the drawings and submit formal drawings herewith.

Priority Claim

As requested by the Examiner, Applicants have amended the specification to include the priority claim. The claim for priority was first made when the application was filed on February 9, 1999, as acknowledged by the Patent Office in the filing receipt.

Objection to the Brief Summary of the Invention

Applicants acknowledge the Examiner's objection to the Brief Summary of the Invention, and have amended the summary accordingly. The new summary finds support, for example, at Page 7, line 3 through page 9, line 28, and Figures 3, 4, and 5 of the Application.

Cited Art

U.S. Patent No. 5,261,080 to Khoyi et al. ("Khoyi") is entitled "Matchmaker for Assisting and Executing the Providing and Conversion of Data Between Objects in a Data Processing System Storing Data in Typed Objects Having Different Data Formats," and U.S. Patent No. 5,280,610 to Travis et al. ("Travis") is entitled "Methods and Apparatus for Implementing Databases to Provide Object-Oriented Invocation of Applications."

Patentability of Claims 71-79, 81-83, 27, 30-34, and 49 Over Khoyi under § 103

The Action rejects claims 71-79, 81-83, 27, 30-34, and 49 under 35 U.S.C. § 103(a) as unpatentable over Khoyi. Applicants respectfully submit the claims in their amended form are

allowable over the cited art. To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. (MPEP § 2142.) In the present case, the references fails to teach or suggest at least one claim limitation.

Claim 71

Claim 71 is directed to a method for manipulating an object displayable in a client application and recites in part:

> routines of the application programming interface are divided into an object-independent client library and a server library, the object-independent client library comprising routines which invoke the proper server application to manipulate the object, and the server library comprising routines which process requests to manipulate the object;
>
> ...
>
> invoking by the server library the server application to perform the requested manipulation of the object.

For example, the Application at FIG. 3 shows a client application, a client library, a server library, a server application, and a containee object.

Claim 71 stands rejected over Khoyi. Specifically, the Action relies on Khoyi's description of object managers, including Khoyi's description at column 10, lines 14-18, which state:

> The system of the present invention provides for a plurality of object managers to operate with any given object type and certain object managers, for example, certain utilities, may operate with more than one type of object.

Thus, Khoyi does describe "a plurality of object managers to operate with any given object type." However, there are significant differences between Khoyi and the recited arrangement.

Khoyi's description of object managers would not lead one to the recited arrangement, which includes, "application programming interface routines divided into an object-independent library and a server library" and "invoking by the server library the server application to

perform the requested manipulation of the object." Khoyi describes its object managers at column 9, lines 61-63, as follows:

> Programs for operating upon objects are known as "object managers" or are sometimes referred to as "editors", "applications programs", or applications".

Khoyi's object managers therefore would not lead one to the recited arrangement of invoking by the server *library* the server *application* to perform the requested manipulation of the object.

For at least these reasons, claim 71 and its dependent claims, 27-49, 72-77, 82 and 84-88, are allowable over Khoyi.

Claim 78

Claim 78 is directed to a method for manipulating an object within a container object and recites in part:

> providing an application programming interface supporting embedded or linked objects wherein the application programming interface is separated into an object-independent client library and a server library, the object-independent client library comprising routines which determine servers to manipulate embedded or linked objects, and the server library comprising routines which invoke server routines on embedded or linked objects;
> ...
> invoking by the server library at the determined server, a server routine to create the selected embedded or linked object;

For example, the Application at FIG. 3 shows a client application, a client library, a server library, a server application, and a containee object.

Claim 78 stands rejected over Khoyi. Specifically, the Action relies on Khoyi's description of object managers, including Khoyi's description at column 10, lines 14-18, which state:

> The system of the present invention provides for a plurality of object managers to operate with any given object type and certain object managers, for example, certain utilities, may operate with more than one type of object.

Thus, Khoyi does describe "a plurality of object managers to operate with any given object type." However, there are significant differences between Khoyi and the recited arrangement.

Khoyi's description of object managers would not lead one to the recited arrangement, which includes, "application programming interface ... seperated into an object-independent

client library and a server library" and "invoking by the server library at the determined server, a server routine to create the selected linked or embedded object." Khoyi describes its object managers at column 9, lines 61-63, as follows:

> Programs for operating upon objects are known as "object managers" or are sometimes referred to as "editors", "applications programs", or applications".

Khoyi's object managers therefore would not lead one to the recited arrangement of invoking by the server *library* the server *application* to create the selected object.

For at least these reasons, claim 78 and its dependent claims, 79, and 83 are allowable over Khoyi.

Claim 81

Claim 81 is directed to means for displaying objects within a client means and recites in part:

> client library means for receiving from the client means, requests to perform manipulations on containee objects, consulting the configuration store means and determining an appropriate server means out of the plural server means to perform manipulations based on object class identifiers, and sending messages to server library means associated with determined server means, said messages comprising requests to perform manipulations on containee objects; and
>
> server library means for receiving messages to perform requested manipulations, and invoking server routines for performing requested manipulations.

For example, the Application at FIG. 3 shows a client application, a client library, a server library, a server application, and a containee object.

Claim 81 stands rejected over Khoyi. Specifically, the Action relies on Khoyi's description of object managers, including Khoyi's description at column 10, lines 14-18, which state:

> The system of the present invention provides for a plurality of object managers to operate with any given object type and certain object managers, for example, certain utilities, may operate with more than one type of object.

Thus, Khoyi does describe "a plurality of object managers to operate with any given object type." However, there are significant differences between Khoyi and the recited arrangement.

Khoyi's description of object managers would not lead one to the recited arrangement, which includes, "client library means for consulting the configuration store means and determining an appropriate server means out of the plural server means to perform manipulations and sending messages to server library means associated with determined server means" and a "server library means for receiving messages to perform requested manipulations; and invoking server routines for performing requested manipulations." Khoyi describes its object managers at column 9, lines 61-63, as follows:

> Programs for operating upon objects are known as "object managers" or are sometimes referred to as "editors", "applications programs", or applications".

Khoyi's object managers therefore would not lead one to the recited arrangement of invoking by the server *library* the *server routines* to perform the requested manipulation of the object.

For at least these reasons, claim 81 and its dependent claims, 79, and 83 are allowable over Khoyi.

Patentability of Claims 27-29 and 35-49 Over Khoyi in view of Travis under § 103

The Action rejects claims 27-29 and 35-49 under 35 U.S.C. § 103(a) as unpatentable over Khoyi in view of Travis. Because the claims rejected over Khoyi and Travis all depend from claim 71, Applicants respond by pointing out that a Khoyi-Travis combination does not teach or suggest the arrangement of claim 71. Therefore, claim 71 and its rejected dependent claims 27-29 and 35-49 are allowable over Khoyi in light of Travis.

Claim 71 recites in part:

> routines of the application programming interface are divided into an object-independent client library and a server library, the object-independent client library comprising routines which invoke the proper server application to manipulate the object, and the server library comprising routines which process requests to manipulate the object;
>
> ...
>
> invoking by the server library the server application to perform the requested manipulation of the object.

In the rejection of dependent claims 27-29 and 35-49, the Action relies on Travis' description at column 24, lines 13-24, which states:

> Next, the invoker software component 1236 transmits a query to the control server software component 1334 of the

> preferred server platform which causes control server software component 1334 to query a control server registry 1425 to determine whether the desired method server on the preferred server platform is available to process the method identified in the processed method invocation request. Availability of a method server is determined in the preferred implementation by examining in the control server registry 1425 to find out whether the method server is currently able to process methods invoked by client applications.

Thus, Travis does describe "an invoker software component that transmits a query to a control server software component that queries a control registry to determine if a preferred server is available." However, there are significant differences between Travis and the recited arrangement.

Travis' description of an invoker software component would not lead one to modify Khoyi to arrive at the recited arrangement, which includes, "application programming interface routines divided into an object-independent library and a server library" and "invoking by the server library the server application to perform the requested manipulation of the object." For example, Travis describes that before querying a control server, the invoker software component must (1) query a context object database to determine a method identifier (column 23, lines 47-50), (2) query a server registration facility to find a server platform for the method identifier (column 23, lines 52-63), (3) query a context database for the location of a running server (column 23, line 58), and query the control software component causing it to query a control server registry to find the server on the preferred platform (column 24, lines 14-15); ... all before invoking a method on a dispatcher software component (column 24, lines 29-31). However, none of these would lead one to modify Khoyi to arrive at the recited "application programming interface routines divided into an object-independent library and a server library" and "invoking by the server library the server application to perform the requested manipulation of the object." For at least these reasons, claim 71 and its dependent claims 27-29 and 35-49 are allowable over a Khoyi-Travis combination.

New Claims

New claims 84-88 recite various patentably distinct subject matter not taught or suggested by Khoyi in view of Travis. For example, claim 87 recites "the client library and the

server library send messages via a channel comprising inter-processes communication." Khoyi in light of Travis fails to teach or suggest such an arrangement. For at least these reasons, claims 84-89 are allowable over a Khoyi-Travis combination. Such action is respectfully requested.

Request For Interview

If any issues remain, the Examiner is formally requested to contact the undersigned attorney prior to issuance of the next Office Action in order to arrange a telephonic interview. It is believed that a brief discussion of the merits of the present application may expedite prosecution. Applicants submit the foregoing formal Amendment so that the Examiner may fully evaluate Applicants' position, thereby enabling the interview to be more focused.

This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

Conclusion

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By _____
Gregory L. Maurer
Registration No. 43,781

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446

Marked-up Version of Changes to Specification and Claims

In the Specification:

The following has been inserted at page 1, line 8 of the specification:

RELATED APPLICATION DATA

This application is a continuation of application No. 08/884,448 filed June 27, 1997 and now issued as U.S. Patent No. 5,905,884, which is a continuation of application No. 08/382,214, filed January 30, 1995 and now issued as U.S. Patent No. 5,692,157, which is a continuation of application No. 07/900,968, filed June 17, 1992, abandoned.

The Section with the heading "SUMMARY OF THE INVENTION", at page 3, line 31, has been amended as follows:

SUMMARY OF THE INVENTION

The present technology is directed to providing facilities to manipulate containee objects within container objects. In one respect, an application that creates a container object is called a client, and an application that creates or manipulates a containee object is called a server. For example, a client can select among the various containee objects within the container object, and request via a client library, a proper server to manipulate the selected containee object. A server can manipulate the contents of a containee object in response to a request received from the client via a server library.

In another respect, clients and servers can be provided with implementation-independent client libraries and server libraries, respectively, that provide object linking and embedding functionality between clients and servers. The client and server libraries provide a set of routines that manage, among other things, the setup and initialization for clients to send and receive messages and data to and from servers. The client library provides routines which invoke the correct server application to act upon a particular containee object. The server library provides routines which process requests to manipulate containee objects.

In one implementation, client applications and server applications are separate processes. The client process includes client application code, a client library, and a container object. The server process includes server application code, a server library, and containee objects. In such an implementation, a client library can communicate with a server library through a communications channel. The client library and server library can be dynamically linked with the client application code and server application code, respectively, when an application process is started up. However, the client library and server library could be implemented as processes separate from the client and server processes.

Additional features and advantages will be made apparent from the following detailed description of the illustrated embodiment which proceeds with reference to the accompanying drawings.

[Summary of the Invention

It is an object of the present invention to provide a method and system for registering data formats for a server application.

It is another object of the present invention to provide a method and system for registering data formats that the server application can both receive and send data in.

It is another object of the present invention to provide a method and system for a client application to determine which data formats a server application supports without launching the server application.

It is another object of the present invention to provide a method and system for transferring data between a server and client application.

These and other objects, which will become apparent as the invention is more fully described below, are obtained by a method and system for transferring data between a server and client application. In a preferred embodiment, the data formats that the server application supports are stored in a persistent global registry. The client application retrieves the data formats from the persistent global registry and requests the server application to supply data in the retrieved format. The server application, upon receiving the request, supplies the data in the retrieved format.]

The paragraph at page 11, line 19, has been amended as follows:

In addition to the client and server libraries, the object linking and embedding facilities of the present invention provide information to client and server applications through a persistent global "registry." This registry is a database of information such as (1) for each type of object, the server application that implements the object type, (2) the actions that [the] each server application provides to client applications, (3) where the executable files for each server application are located, and (4) whether each server application has an associated object handler.

The paragraph at page 11, line 30, has been amended as follows:

Communication between client and server processes occurs either synchronously or asynchronously. Synchronous communication occurs when one process sends a message to the other process and the sending process suspends activity until the other process completely processes the message. For example, when a client process wants to create a new containee object, the client process (through the client library routines) sends a message to the appropriate server process. The client process waits until the containee object is created before continuing execution. Asynchronous communication occurs when one process sends a message to the other process and the sending process continues execution while the receiving process responds to the message. Typically, when the receiving process has completed responding to the request, it sends a message indicating such completion to the sending process. For example, when a client process wants a containee object saved in a compound document file, the client process sends a message to the server process. The client process can continue to execute (e.g. responding to users requests) while the server process is saving the object. When the server process has completed saving the object, it sends a completion message to the client process.

The paragraph at page 35, line 13, has been amended as follows:

Although the present invention has been described in terms of a preferred embodiment, it is not intended that the invention be limited to this embodiment. Modifications within the spirit

of the invention will be apparent to those skilled in the art. The scope of the present invention is defined by the claims which follow.

In the Claims:

Claims 27-48, 71-75, 78, 79, and 81 have been amended, claims 49, 76, 77, 82, and 83 remain unchanged, and claims 84-88 are new.

27. (**Twice** Amended) The method of claim 71 wherein the client **library** determines the server **application** based on an association with the class identifier.

28. (**Twice** Amended) The method of claim 27 wherein the association is recorded during installation of the server **application**.

29. (**Twice** Amended) The method of claim 27 wherein the association is recorded when the server **application** is launched.

30. (**Once Amended**) The method of claim 27 including when the server **application** supports a**n object** [data format] that is compatible with the client **application**, launching the server **application**.

31. (**Once Amended**) The method of claim 30 wherein the client **application** is executing in a process and the server **application** is launched in a separate process.

32. (**Once Amended**) The method of claim 30 wherein the client **application** is executing in a process and the server **application** is launched in the same process.

33. (**Once Amended**) The method of claim 30 wherein the client **application** and the server **application** exchange data using a compatible format.

34. (**Twice** Amended) The method of claim 27 wherein the client **library** determines the association while the server **application** is not executing.

35. (**Twice** Amended) The method of claim 71 wherein the server **application** records in the configuration store, an association between itself and the class identifier.

36. (**Once Amended**) The method of claim 35 including when the server **application** supports a data format that is compatible with the client **application**, launching the server **application**.

37. (**Once Amended**) The method of claim 36 wherein the client **application** is executing in a process and the server **application** is launched in a separate process.

38. (**Once Amended**) The method of claim 36 wherein the client **application** is executing in a process and the server **application** is launched in the same process.

39. (**Once Amended**) The method of claim 36 wherein the client **application** and the server **application** exchange data using a compatible format.

40. (**Twice** Amended) The method of claim [36]71 wherein the client **application, the client library, the server application, and the server library, each execute in a separate process** [determines the data formats while the server is not executing].

41. (**Twice** Amended) The method of claim 71 for supplying the server **application** to perform the requested manipulation wherein the server **application** populates the configuration store with class identifiers it supports.

42. (**Twice** Amended) The method of claim 41 wherein the client **application** determines the server **application** that supports the class identifier while the server **application** is not executing.

43. (**Twice** Amended) The method of claim 41 wherein the server **application** populates the configuration store during installation of the server **application**.

44. (**Twice** Amended) The method of claim 41 wherein the server **application** populates the configuration store when the server **application** is launched.

45. (**Twice** Amended) The method of claim 41 including when the server **application** supports a data format that is compatible with the client **application**, launching the server **application**.

46. (**Once Amended**) The method of claim 45 wherein the client **application** is executing in a process and the server **application** is launched in a separate process.

47. (**Once Amended**) The method of claim 45 wherein the client **application** is executing in a process and the server **application** is launched in the same process.

48. (**Once Amended**) The method of claim 45 wherein the client **application** and the server **application** exchange data using a compatible format.

49. (Amended) A computer-readable medium containing instructions for causing a computer system to perform the method of claim 71.

71. (**Twice** Amended) A method in a [client and a server] computer system for manipulating an object displayable in [the]**a** client **application via any one of a plurality of server applications using an application programming interface**[and identified by a class identifier], the computer system having a configuration store for storing [the] **a** class identifier **associated with the object** and associating the class identifier with a**t least one of the** server **applications** [out] of [a]**the** plurality of server[s] **applications**, the method comprising:

requesting by the client **application** through [an]**the** application programming interface, a manipulation to be performed on the object **wherein routines of the application programming interface are divided into an object-independent client library and a server library, the object-independent client library comprising routines which invoke the proper**

server application to manipulate the object, and the server library comprising routines which process requests to manipulate the object;

determining **by the object-independent client library using** [from] the configuration store **and**[via] the class identifier **of the object**, a server **application** out of a plurality of server[s] **applications** to perform the requested manipulation on the object; [and]

sending **by the object-independent client library,** a message to the server **library** to perform the requested manipulation on the object;

receiving by the server library the message to perform the requested manipulation on the object; and

invoking by the server library the server application to perform the requested manipulation on the object.

72. **(Once Amended)** The method of claim 71, wherein the object displayable in the client **application** is a first object, the method further comprising:

depicting the first object as appearing inside a second object displayable in the client **application**.

73. **(Once Amended)** The method of claim 71, wherein the client **application** determines from the configuration store and displays for a user a list of available manipulations on the object.

74. **(Once Amended)** The method of claim 71, wherein the server **application** is started up in response to receiving the message.

75. **(Once Amended)** The method of claim 71, wherein the server **application** shuts down after completion of manipulations requested in the message.

76. (Unchanged) The method of claim 71 wherein a user can select a new object from amongst a plurality of embedded or linked objects displayed in a graphical user interface.

77. (Unchanged) The method of claim 71 wherein a user can select a manipulation or procedure to be performed on a selected object from amongst a plurality of manipulations or procedures displayed in a graphical user interface.

78. **(Once Amended)** A method in a [client and a server] computer system, the computer system having a configuration store for storing identifiers of available embedded or linked objects and identifiers of servers associated with the embedded or linked objects, the method comprising:

providing an application programming interface supporting embedded or linked objects wherein the application programming interface is separated into an object-independent client library and a server library, the object-independent client library comprising routines which determine servers to manipulate embedded or linked objects, and the server library comprising routines which invoke server routines on embedded or linked objects;

requesting by a user from [the]**a** client **application**, creation of an embedded or linked object;

requesting by the client application via the application programming interface to the object-independent client library, available embedded or linked objects;

determining **by the object-independent client library** from the configuration store [and presenting to the user by the client] **and returning to the client application,** a list of the available linked or embedded objects;

presenting by the client application to the user, the available embedded or linked objects;

selecting by the user from the available **presented objects** [list presented by the client] an object to be linked or embedded within a container object **displayed via the client application**; [and]

requesting by the client application via the application programming interface to the object-independent client library, creation of the selected embedded or linked object;

determining **by the object-independent client library from the configuration store** a server associated with the **selected** linked or embedded object [to implement the selected linked

or embedded object] and sending a message from the **object-independent** client **library** to the server **library,** to create the selected linked or embedded object;

receiving by the server library, the message to create the selected embedded or linked object; and

invoking by the server library at the determined server, a server routine to create the selected embedded or linked object;

whereby the created linked or embedded objected is created by the server **routine** and the user can edit or otherwise manipulate the **created** linked or embedded object **within the container object**.

79. **(Once Amended)** The method of claim 78 wherein the user is able to edit or manipulate a linked or embedded object by selecting an action available on a client **application** menu.

81. **(Once Amended)** A computer software system comprising:

client means for displaying [an] **containee** objects [displayable] within the client means, wherein [the] **containee** objects [is] **are** associated with [an] object class **identifiers**[identifying means];

plural server means for performing manipulations on [the] **containee** objects [displayable within the client means];

configuration store means for storing an association between [the] object class **identifiers and server means that perform manipulations on associated object classes**[identifying means for the object and the server means; and]**;**

client library means for[,] receiving **from the client means,** requests to perform [a] manipulations on [the] **containee** objects [via the client means; receptive to the means for receiving a request, means for]**,** consulting the configuration store means and determining an appropriate server means out of the plural server means to perform [the] manipulations based on [the] object class **identifiers**[identifying means;]**,** and **sending messages to server library means associated with determined server means, said messages comprising requests to perform manipulations on containee objects;** [means for launching the appropriate server means to perform the manipulation on the object.]**; and**

server library means for receiving messages to perform requested manipulations, and invoking server routines for performing requested manipulations.

82. **(Unchanged)** A computer-readable medium containing instructions for causing a computer system to perform the method of claim 27.

83. **(Unchanged)** A computer-readable medium containing instructions for causing a computer to perform the method of claim 78.

84. (New) The method of claim 71 further comprising:

receiving by the server library, an indication from the server application that the requested manipulation is complete;

sending a message from the server library that the requested manipulation is complete;

receiving by the client library, the message from the server library that the requested manipulation is complete; and

sending an indication to the client application that the requested manipulation is complete.

85. (New) The method of claim 71 wherein the client application and the client library are dynamically linked to execute in the same process.

86. (New) The method of claim 71 wherein the server application and the server library are dynamically linked to execute in the same process.

87. (New) The method of claim 71 wherein the client library and the server library send messages via a channel comprising inter-process communication.

88. (New) The method of claim 71 wherein the client application, the client library, the server application, and the server library are processes sharing the same processor.

89. (New) The method of claim 71 wherein the application programming interface provides functions comprising compound document functionality.